MODÜL 2

YORUM SATIRLARI, DEĞİŞKENLER, VERİ TİPLERİ VE OPERATÖRLER

Şekil 2.1: *Bölümle ilgili örnek uygulamalara karekoddan ulaşabilirsiniz.*

2.1. Yorum Satırlarını Kullanma

Yorum satırları, Python yorumlayıcısı tarafından dikkate alınmayan ve yorumlanmayan ifadelerdir. Python'da yorum satırları genel olarak aşağıdaki işlemler için kullanılır:

- Bir hatırlatma eklemek
- Program veya kodlarla ilgili bir açıklama yapmak
- Kullanılmayan bir kod satırını pasif hale getirmek
- Süsleme yapmak

Bu tür açıklama satırları kodun başkaları tarafından daha iyi anlaşılmasını sağlar. Python'da tek satırlık açıklama için "**#**" işareti kullanılır. "#" işareti kullandığınızda o satırdaki metin kod olarak işlenmez.

Örnek 1

Aşağıdaki yorum satırları Python tarafından dikkate alınmamaktadır. Bu nedenle sadece "print()" komutu çalışır.

```
#Bu kod ekrana yazı yazılmasını sağlamaktadır.
print ("Konu: Yorum satırlarını kullanma")
#Her satırın başına # işareti eklenerek
#alt alta yorum satırları oluşturulabilir.
Konu: Yorum satırlarını kullanma
```

Örnek 2

Yorum satırlarını kod satırının devamında aynı satırda kullanabilirsiniz. Bu kullanımda önce kod gelir, devamında yorum satırı "#" işareti ile başlar (öndeki kod çalışır), daha sonra satır sonuna kadar yorum satırı olarak dikkate alınmaz.

```
print (2+3) # Bu kod satırı ekrana 2 sayının toplamını yazar.
5
```

Her satırın başına "#" işareti ekleyerek alt alta yorum satırları oluşturabilirsiniz. Yorum satırlarında özel karakterler etkisizdir.

Örnek 3

Birden fazla yorum satırı kullanılacaksa yorumlar üçlü tek tırnak veya üçlü çift tırnak blokları arasına yazılır.

```
'''Python'da birden fazla açıklama satırı kullanmak için üçlü tek tırnak veya çift
tırnak kullanılır. Açıklama satırını bitirmek için aynı işaretler kullanılır.'''
'Python'da birden fazla\naçıklama satırı\nkullanmak için üçlü tek tırnak veya çift\n
tırnak kullanılır. Açıklama satırını bitirmek için \naynı şekilde kullanılır.'
```

Konsol çıktısında açıklamalar birden fazla satırda olduğu için satır sonlarına \n kaçış dizisi karakterini konsol otomatik olarak eklemiştir. Aynı yorum satırını """yorum satırları""" üçlü çift tırnak kullanarak da yapılabilir. Yorum satırları içinde kaçış dizisi karakterlerinin de çalışmadığı unutulmamalıdır.

Örnek 4

Yorum satırlarını Python yorumlayıcı dikkate almaz. Ancak aşağıdaki kod satırları yorum satırı olarak değerlendirilmez.

```
#!/usr/bin/env python3 veya #!c:/Python/python.exe
# -*- coding: utf-8 -*-
"#!/usr/bin/env python3" kodu Python 3 yorumlayıcısının Linux için dosya konumunu
belirtir.
"#!c:/Python/python.exe" kodu Python 3 yorumlayıcısının Windows için dosya konumunu
belirtir.
"# -*- coding: utf-8 -*- " Kullanacağınız karakter kodlamasını belirtmek için
kullanılır. utf-8 Türkçe alfabeyi de destekleyen bir karakter kodlama sistemidir.
```

Örnek 5

Yorum satırları süsleme amacıyla da kullanılabilir. Bazı program dosyalarında aşağıdaki gibi süslü açıklamalar, etiketler görülebilir.

```
'''
#########################################################################
#*********************************************************************##
#                        Python Öğreniyorum                           ##
#                             Python 3                                ##
#*********************************************************************##
#########################################################################
'''
```

2.2. Değişkenler

Kod yazarken sadece sabit değerler üzerinden işlemler yapılmaz. Kullanıcıdan veya başka kaynaklardan veri alınması gerekir. Örneğin, kullanıcıya ismiyle "merhaba" diyeceğimiz bir kod yazmak istersek her kullanıcıdan ismini girdi olarak almamız gerekir. İşlem yapabilmek için bu girdiler (değerler) bellekte tutulmalıdır. Bu girdileri depolamak amacıyla bellekte belirli bir yer ayrılması gerekir. Bir değişken tanımlandığında yorumlayıcı, veri türüne bağlı olarak bellekte yer ayırır ve ayrılan bölümde hangi türden verilerin saklanabileceğini belirler. Değişkenler bir isim, sayı veya farklı türdeki bir veri için bellekte ayrılan bu yeri temsil eder. Değişkenlere farklı veri tiplerinde değerler atanabilir. Değer atama ile (bir değişkeni bir değere eşleyerek) tam sayı, ondalık sayı, dizi veya karakter dizisi türünde değerler değişkenlerde tutulabilir. Python, bu konuda çok esnektir. Python'da değişkenlerinin veri tiplerini açıkça bildirmeye gerek yoktur. Aynı değişken farklı veri tiplerinde değerler alabilir. Aynı değişkene önce sayı, sonra bir metin daha sonra başka türde bir değer atanabilir. Bir değişkene değer atandığında veri tipi otomatik olarak tanımlanır. Eşittir operatörü "=" değişkenlere değer atamak için kullanılır. "=" Operatörünün solunda değişkenin adı ve "=" operatörünün sağında ise bu değişkene atanacak değer yer alır.

Örnek 6

Değişken oluşturma ve değer atamaya ilişkin örnekler:

```
#sayi1 değişkenine 5 sayısı atandı.
sayi1=5
print('Değişkenin içindeki sayı: ', sayi1)
sayi1=10
print('Değişkenin içindeki sayı: ', sayi1, 'oldu')
sayi1='Murat'
print ('Değişkenin içindeki değer: ', sayi1, 'oldu')
sayi1=10.5
print ('Değişkenin içindeki sayı: ', sayi1, 'oldu')
Değişkenin içindeki sayı:  5
Değişkenin içindeki sayı:  10 oldu
Değişkenin içindeki değer:  Murat oldu
Değişkenin içindeki sayı:  10.5 oldu
```

Örnekte aynı değişkene hem karakter dizisi (string) hem de sayısal değerler atanmıştır.

Bir değişkene değer atanırken birden fazla yöntem kullanılabilir. Örnekte buna ilişkin kodlar verilmiştir.

Örnek 7

Aşağıda 3 değişkene de tek satırda 1 değeri atanmıştır.

```
a = b = c = 1
print ('1. sayı=', a)
print ('2. sayı=', b)
print ('3. sayı=', c)
1. sayı= 1
2. sayı= 1
3. sayı= 1
```

Örnek 8

Değişkenler aralarına virgül eklenerek yan yana yazılır. Değerleri de aynı sıralama ile karşılarına yazılır.

```
adi, soyadi, yasi='Canan', 'DAĞDEVİREN', 34
print ("Adı=", adi)
print ("Soyadı=", soyadi,)
print ("Yaşı=", yasi)
Adı= Canan
Soyadı= DAĞDEVİREN
Yaşı= 34
```

Örnek 9

Değişkenlere değer atamak için başka bir yöntem aralarına noktalı virgül ";" ekleyerek değişken - değer ikilileri şeklinde yazmaktır.

```
adi='Aziz'; soyadi='SANCAR'; yasi=73
print ("Adı=", adi)
print ("Soyadı=", soyadi,)
print ("Yaşı=", yasi)
Adı= Aziz
Soyadı= SANCAR
Yaşı= 73
```

Örnek 10

Değer atanmayan ve/veya tanımlanmamış bir değişken kullanılırsa Python hata verir.

```
print (yenisayi)
#Python değişkenleri değer atandığında tanımlandığı için hata mesajı alırsınız.
# yenisayi değişkeni tanımlanmamış olduğu için hata mesajı alınır.
NameError                 Traceback (most recent call last)
<ipython-input-1-9ff08615337f> in <module>()
----> 1 print (yenisayi)

NameError: name 'yenisayi' is not defined
```

Değer atamadan tanımlamak için yenisayi=int( ) kodu kullanılabilir. Bu durumda değişkene ilk değer olarak "0" sıfır atanır.

```
yenisayi=int( )
print(yenisayi)
0
```

Örnek 11

Değişkenler veri tiplerine göre kullanılmazsa Python hata verir.

```
sayi1=1
metin1='deneme'
print(sayi1+metin1) #Bir sayı ile bir metin, kelime toplanamaz.
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-47-e585c633881d> in <module>()
      1 sayi1=1
      2 metin1='deneme'
----> 3 print(sayi1+metin1) #Bir sayı ile bir metin, kelime toplanamaz.

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

2.2.1. Değişken Adlandırmada Kurallar

Değişken adı verilirken uyulması gereken bazı kurallar ve kurallar kadar katı olmasa da yararlı kullanım önerileri vardır.

Değişken adlandırma kuralları:

- Değişkenler bir harf (a - z, A - Z) veya alt çizgi (_) ile başlamalıdır. Bunların dışında sayı veya başka bir karakter ile de başlayamaz.
- Değişken adında rakam, alt çizgi(_), büyük veya küçük harf olabilir.
- Değişken adları herhangi bir uzunlukta olabilir.

Python, değişken adlandırmada Türkçe karakterlerin (ç , ğ, ı, ö, ş ve ü) kullanımına izin vermektedir. Ek olarak Python programlama dilinde ayrılmış sözcükler kullanılamaz. Bu özel sözcüklerin listesini görmek için aşağıdaki kod kullanılabilir.

```
import keyword
keyword.kwlist
['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', 'continue',
'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global',
'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass',
'raise', 'return', 'try', 'while', 'with', 'yield']
```

Büyük harf ve küçük harf kullanılarak tanımlanan değişkenlerin adı aynı olsa bile farklı değişkenler olduğu unutulmamalıdır.

Örnek 12

Python dilinde uygun değişken adı örnekleri:

```
#Uygun değişken isimleri
sayi1=1
Sayi1=2
```

Sayi1' ekrana yazdırılırsa çıktı ne olur?

```
print (sayi1)
print(Sayi1)
#Büyük harf ve küçük harf kullanarak tanımlanan değişkenlerin adı aynı olsa bile
farklı değişkenler olduğunu unutulmamalıdır.
sayı1=3
#Python  değişken adlandırmada Türkçe karakter kullanımına izin vermektedir.
print(sayı1)
1
2
3
```

Örnek 13

Python'da hatalı değişken adı kullanımı örneği:

```
1sayi=5 #Hatalı değişken adı.
print (1.sayi)
  File "<ipython-input-52-a0b35430cdb5>", line 1
    1sayi=5 #Hatalı değişken adı.
       ^
SyntaxError: invalid syntax
```

2.2.2. Değişken Adlandırma için Standartlar

Değişken adlandırma için bazı standartlar vardır. Bu standartlar değişken adının ve içeriğinin anlaşılmasına yardımcı olarak programcıların daha kolay çalışmasını sağlar. Değişken adı, onun içeriği hakkında bilgi verirse kodun anlaşılması kolaylaşır. Değişken adlarının yazımında bazı standart kullanımlar vardır. Birden fazla kelimenin kullanılacağı değişken adlarında kelimelerin ilk harfi büyük olabilir. Camel standardında (başka standartlar da bulunmaktadır) değişken adlarının görünüşü deve hörgücüne benzetilmektedir. Değişkenin adına küçük harfle başlanır ve sonraki her kelime büyük harfle başlar.

Örnek 14

```
adi= "Elif"
soyadi="Altun"
dogumYili=1981
universiteMezunuMu=True
universiteyeBasladigiYil=1999
mezuniyetNotu=2.00
print ('Adı: ', adi)
print ('Soyadı: ', soyadi)
print('Üniversite Mezunu mu? ', universiteMezunuMu)
print('Üniversiteye Başladığı Yıl: ', universiteyeBasladigiYil)
print('Mezuniyet Notu: ', mezuniyetNotu)
Adı:  Elif
Soyadı:  Altun
Üniversite Mezunu mu?  True
Üniversiteye Başladığı Yıl:  1999
Mezuniyet Notu:  2.0
```

2.3. Veri Tipleri

Veri tiplerini anlayabilmek için aşağıdaki kod satırı incelenebilir. Aşağıda bir sayı ile bir karakter dizisi üzerinde operatörleri kullanarak işlemler yapılmıştır.

Örnek 15

Hatalı veri tipi kullanımına örnek: "sayi2" değişkeninin tek tırnak içinde verildiğine ve bir karakter dizisi olduğuna dikkat edilmelidir.

```
sayi1=5
sayi2='3'
print (sayi1+sayi2)
TypeError                          Traceback (most recent call last)
<ipython-input-55-294ee141ba94> in <module>()
      1 sayi1=5
      2 sayi2='3'
----> 3 print (sayi1+sayi2)
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Kod çalıştırıldığında bir hata mesajı alınır. Bir aritmetik operatörü kullanılırken bir sayı ile bir karakter dizisi (ikinci değişkenin adı sizi yanıltmasın) toplamaya çalışıldığı için Python hata verir. Veri tipleri, değişkenler üzerinde yapılabilecek işlemleri belirler.

Örnek 16

Operatörler konusunda * operatörü ile ilgili bir ayrıntıdan bahsedilmiştir.

```
sayi1=5
sayi2='3'
#sayi2 değişkenin tek tırnak içinde verildiğinde bir karakter dizisi olduğuna dikkat
ediniz.
print (sayi1*sayi2)
#Sizce nasıl bir sonuç çıkar?
33333
```

Örnekte görüldüğü gibi kod "çarpma" işlemi yapamamıştır. Çünkü ortada iki sayısal değer yoktur. İkinci örnekte 3 sayısını bir karakter olarak 5 defa yazmıştır.

Python'da değişkenlere değer atanırken veri tipleri belirtilmez. Python, atanan değere göre veri türünü kendisi belirler. Ancak programlama yaparken veri tiplerini bilmek ve ona göre kullanmak gerekir.

2.3.1. Python'da Sık Kullanılan Veri Tipleri

Python programlama dilinde sıkça kullanılan veri tipleri aşağıdaki tabloda sunulmuştur:

Veri Tipi	Sınıfı	Açıklama
Integer	int	Tam sayı. (Örnek: 3, 5, 369963)
Float	float	Ondalıklı sayı. (Örnek: 10.45)
Complex	complex	Karmaşık sayılar A + Bj şeklinde kullanılır. (Örnek: 4+5j)
Karakter dizisi (String)	str	Karakter dizilerini (metinleri) göstermek için kullanılır. Çift tırnak veya tek tırnak içinde gösterilir. "Merhaba Dünya"
Boolean	bool	Sadece True veya False değeri alır. int(True)=1 iken int(False)=0 dır.
Liste	list	Farklı veri türleri içerebilir. listem=['Çınar', 24, 'Mühendis', True]
Demet (tuple)	tuple	Farklı veri türleri içerebilir. demet1=('Çınar', 24, 'Mühendis', True)
Sözlük (dictionary)	dict	Farklı veri türleri içerebilir. sozluk={'adi': 'Çınar','yasi'=24, 'meslekUnvani':'Mühendis', 'askerlikDurumu': True}

2.3.2. Sayısal (int, float ve complex) Veri Tipleri

"int" veri tipi Python'da tam sayıların tutulduğu veri tipidir: 3, 5, 198763 gibi değerleri tutar. En çok kullanılan veri tiplerinden biridir.

"float" veri tipi ondalıklı sayıların tutulduğu veri tipidir: 0.5, 234678.67 gibi değerleri tutar.

"complex" veri tipi ise karmaşık sayıların tutulduğu veri tipidir. A+Bj tipinde veriler tutulur: 4+5j gibi değerleri tutar.

Örnek 17

Python'da bir değişkene değer atandığında veri tipleri atanan değere göre otomatik olarak belirlenir.

```
piSayisi=3.14
#float tipinde bir veri
print ('pi sayısı=', piSayisi)
rCm=2
#integer tipinde veri
alan=3.14*2**2
print ('Alan=', alan)
#sonuç float tipinde
print('Yarıçapı 2 olan dairenin alanı ', alan, ' cm 2 dir' )
karmasikSayi=4+5j
print('Bir karmaşık sayı=', karmasikSayi+3j)
pi sayısı= 3.14
Alan= 12.56
Yarıçapı 2 olan dairenin alanı  12.56  cm 2 dir
Bir karmaşık sayı= (4+8j)
```

2.3.3. Karakter Dizisi (string) Veri Tipi

Karakter dizisi, kullanıcıdan alınan değerlerin metin formatında tutulduğu veri tipleridir. Python karakter dizisi oldukça kullanışlı işlevlere sahiptir.

Örnek 18

Bir karakter dizisi ekrana yazdırılabilir, başka bir karakter dizisiyle birleştirilebilir.

"len( )" metodu bir karakter dizisinin uzunluğunu vermektedir.

```
metin1 = 'Merhaba '
metin2 = 'Mars'
print (metin1)            # karakter dizisinin tamamını yazar
print (metin1 * 2)        # karakter dizisini 2 defa yazar
print (metin1 + metin2)   # iki karakter dizisini birleştirir
print ('metin1  adlı değişkendeki değerin uzunluğu:', len(metin1))
#Boşluğun da bir karakter olduğunu gözden kaçırmayınız.
Merhaba
Merhaba Merhaba
Merhaba Mars
metin1 adlı değişkendeki değerin uzunluğu: 8
```

2.3.3.1. Karakter Dizilerinde (string) Dilimleme İşlemleri

Bir karakter dizisinin içindeki karakterlere tek tek veya belirli bir aralıkta erişilebilir. Köşeli parantez içinde [ ] tek bir sayı verildiğinde bu karakter dizisinin indisini ifade eder. İndis numarası "0" dan başlayarak karakterin metindeki kaçıncı sırada yer aldığını gösterir. metin[0] ifadesi metin değişkenindeki 1. karakteri verir. Belirli bir aralıktaki karakteri alırken "[başlangıç indisi:bitiş indisi]" şeklinde ifade edilir. metin[0:5] metin değişkeninde indisi 0, 1, 2, 3 ve 4 olan karakterleri dilimler. Bitiş indisi dilimlemeye dahil edilmez. İndislerin "0" dan başladığı unutulmamalıdır.

Başlangıç indisi verilmeyen karakter dizisinde örnek: metin[:7] başlangıç indisi otomatik olarak sıfır (0) olur. Örnek 0, 1, 2, 3, 4, 5 ve 6. karakterlerden oluşan bir metin verir. Bitiş indisi değeri verilmezse başlangıç indisinden başlanarak son karakter dâhil dilimleme işlemi yapılır.

Negatif İndis Sayıları: Karakter dizisine sondan başlandığını ifade eder. metin[-1] karakter dizisinin en sonundaki karakteri verir. metin [-2] ise sondan ikinci karakteri verir.

Karakter dizilerinde indisler ritmik atlanarak dilimleme yapılabilir. Bunun için [başlangıç indisi: bitiş indisi: ritmik artış miktarı] şeklinde kullanılır. "metin[0:8:2]" : 0'dan başlayarak 0, 2, 4 ve 6 indis numaralı karakterleri dilimler.

Örnek 19

Karakter dizilerinin kullanımı ile ilgili örnekler:

```
metin='Merhaba Mars'
print (metin[0])    # ifadenin ilk karakterini yazar.
print (metin[4:7])  # ifadenin 5, 6 ve 7. karakterlerini yazar.
print (metin[8::])  # 9. karakterden sonuncu karaktere kadar yazar.
print (metin[-2])   # karakter dizisinin en sondan ikinci karakterini yazar.
print (metin [:7])  # indisi 0' dan 7'ye kadar olan (7 dahil değil) karakterleri yazar.
print (metin[8:])   # başlangıç indisinden sonra tüm karakterleri yazar.
print(metin[0:8:2]) # 0, 2, 4 ve 6 indis numaralı karakterleri dilimler.
M
aba
Mars
r
Merhaba
Mars
Mraa
```

Liste, demet ve sözlükler kapsamlı konular olduğu için ayrı bölümlerde verilmiştir.

2.4. type( ) Metodu Kullanımı

Python, her ne kadar veri tiplerini otomatik olarak verse de bir değişkenin veri tipini kontrol etmek ve kullanım amacına göre değiştirmek gerekebilir. Bir değişkenin veri tipini öğrenmek için "type( )" komutu kullanılır.

Örnek 20

Bir değişkenin veri tipi type( ) komutuyla kontrol edilir. type (degiskenAdi) şeklinde yazılır.

```
sayi1=5
sayi2=10.556
metin1="1"
#metin1 değişkenine tırnak içinde verilen sayının string tipinde bir değişken
olduğuna dikkat ediniz.
sayi3=4+5j
askerlikYaptiMi=True
#True doğru, 1, evet anlamındadır.
print ("1. değişkenin veri tipi: ", type(sayi1))
print ("2. değişkenin veri tipi: ", type(sayi2))
print ("3. değişkenin veri tipi: ", type(metin1))
print ("4. değişkenin veri tipi: ", type(sayi3))
print ("5. değişkenin veri tipi: ", type(askerlikYaptiMi))
listem=['Çınar', 24, 'Mühendis', True]
print ("6. değişkenin veri tipi: ", type(listem))
demet1=('Çınar', 24, 'Mühendis', True)
print ("7. değişkenin veri tipi: ", type(demet1))
sozluk={'adi': 'Çınar','yasi': 24, 'meslekUnvani':'Mühendis', 'askerlikDurumu': True}
print ("8. değişkenin veri tipi: ", type(sozluk))
1. değişkenin veri tipi:  <class 'int'>
2. değişkenin veri tipi:  <class 'float'>
3. değişkenin veri tipi:  <class 'str'>
4. değişkenin veri tipi:  <class 'complex'>
5. değişkenin veri tipi:  <class 'bool'>
6. değişkenin veri tipi:  <class 'list'>
7. değişkenin veri tipi:  <class 'tuple'>
8. değişkenin veri tipi:  <class 'dict'>
```

2.5. Veri Tiplerini Dönüştürmek

Tam sayılarla işlem yapıldığında "integer", kesirli sayılarla işlem yapıldığında "float" veri tipi kullanılmaktadır. Değerler üzerinde işlem yaparken (örneğin "input" ile kullanıcıdan veri alırken) içinde sadece rakamlar bulunan "string" ifadeyi sayısal veri tipine dönüştürmek, bazen de bunun tersini yapmak gerekebilir. Bunun için bazı fonksiyonlar bulunmaktadır. Veri tipini dönüştürmek için kullanılan temel fonksiyonlar şunlardır:

int( ) : Veri tipini integer'a çevirir.

float() : Veri tipini float'a çevirir.

str() : Veri tipini karakter dizisine çevirir.

"integer" tipinde bir sayıyla "float" tipinde bir sayı çarpıldığında sonuç "float" tipinde olacağı için Python bu veri tipini otomatik olarak belirler.

Örnek 21

Aşağıdaki örnekte iki sayının da tırnak içinde verilmiş olduğuna dikkat ediniz.

```
metin1='5'
metin2='3'
print (metin1+metin2)
53
```

Yukarıda kullanılan değerler tırnak içinde verildiğinden karakter dizisi veri tipindedir. Kod sonuç olarak iki karakteri yan yana yazacaktır.

Örnek 22

İki değişkenin veri tipini dönüştürünüz. Veri tipi dönüştürüldüğünde karakter dizisi sayıya çevrilmiş olacaktır. Böylece iki sayısal değer üzerinde toplama işlemi yapılabilir.

```
print(int(metin1)+int(metin2))
8
```

Karakter dizisi olan bir değer sayısal bir ifadeye dönüştürüldüğünde bu ifadenin sadece sayısal karakterler içermesi gerekir. “A” harfi veya metinsel bir ifade int(“A”) kullanılarak sayıya çevrilemez.

Veri tipi, kullanılan değerler ile uyumlu olmalıdır. Veri tipi dönüşümünde bazı değerlerde kayıplar olabilir.

Örnek 23

Aşağıdaki örnekte pi değerinin “float” ve “integer” veri tiplerinde kullanıldığında çıkan sonuca dikkat ediniz.

```
piDegeri=3.14
print ('Veri Tipi: ',type(piDegeri))
#3.14 değerini verdiğimizde piDegeri float veri tipi olarak tanımlanacaktır.
yariCap=5
daireninAlani=((piDegeri*2)*yariCap)
print('Dairenin Alanı (float)=', daireninAlani)
piDegeriInt=(int(piDegeri))
# int(piDegeri*2) ifadesi float değeri int veri tipine dönüştürüldü.
print('Int veri tipine dönüştürülen piDegeri: ', piDegeriInt)
daireninAlani=((piDegeriInt*2)*yariCap)
print('Dairenin Alanı (int)=', daireninAlani)
Veri Tipi:  <class 'float'>
Dairenin Alanı (float)= 31.400000000000002
Int veri tipine dönüştürülen piDegeri:  3
Dairenin Alanı (int)= 30
```

Örnek 24

Boolean veri tipini sayısal veri tipine ve string veri tipine dönüştürebilirsiniz. Bu işlemin tersini de yapabilirsiniz.

```
askerlikYaptiMi=True
print('Askerlik yaptı mı?', askerlikYaptiMi)
askerlikYaptiMiInt=int(askerlikYaptiMi) #integer tipine dönüştürüldü.
print('Askerlik yaptı mı?', askerlikYaptiMiInt)
askerlikYaptiMiStr=str(askerlikYaptiMi) #string tipine dönüştürüldü.
print('Askerlik yaptı mı?', askerlikYaptiMiStr)
#Çıktı olarak True verir ancak bu boolean tipinde değildir.
Askerlik yaptı mı? True
Askerlik yaptı mı? 1
Askerlik yaptı mı? True
```

2.6. Operatörler

Python'da aritmetik, mantıksal işlemler ve ilişkisel karşılaştırmalar gibi işlemler yapmak için operatör (işleç) adı verilen semboller ve özel sözcükler kullanılır. Bu bölümde Python'da kullanılan temel operatörler yer almaktadır.

2.6.1. Aritmetik operatörler

Toplama, çıkarma, çarpma ve bölme gibi işlemler başta olmak üzere aritmetik işlemleri yapmak için kullanılan operatörlerdir.

İşaret	İşlem	Örnek	Sonuç
+	Toplama	5+3	8
-	Çıkarma	5-3	2
*	Çarpma	5*3	15
/	Bölme	5/3	1.6666666666666667
**	Kuvvet	5**3	125
//	Tam sayı bölme	5//3	1
%	Mod	5%3	2

Toplama Operatörü

Bu operatör iki veya daha fazla sayısal değeri toplamak için kullanılır.

Örnek 25

Sayısal değerler "integer" , "float" veya "complex" tipinde olabilir.

```
print (5+3)
print (5+3+3)
sayi1=10
print (sayi1+5)
#Aynı şekilde değişken kullanarak da yapabiliriz.
sayi2=10.34
print (sayi1+sayi2+5.5) #farklı veri tiplerindeki sayıları ve değişkenleri de
kullanabiliriz
8
11
15
25.84
```

Örnek 26

Toplama operatörünün karakter dizilerinde farklı bir kullanımı vardır: İki veya daha fazla karakter dizisini birleştirmek için kullanılabilir.

```
print ('Merhaba ' + 'Mars')
metin1='Merhaba ' + 'Mars' + ' nasılsın?'
print(metin1)
Merhaba Mars
Merhaba Mars nasılsın?
```

Çıkarma Operatörü

Bu operatör, sayıların farkını bulmak için kullanılır.

Örnek 27

```
print(5-3)
sayi1=5
sayi2=3
print(sayi1-sayi2)
print (3-4-5)
2
2
-6
```

Çarpma Operatörü

Çarpma operatörü iki veya daha fazla sayıyı çarpmak için kullanılır.

Örnek 28

```
print(5*3)
print(5*3*2)
sayi1=5
sayi2=3
print(sayi1*sayi2)
15
30
15
```

Çarpma operatörü, toplama operatöründe olduğu gibi karakter dizilerinde farklı bir amaç için kullanılmaktadır.

Örnek 29

Bir karakter dizisini istediğiniz kadar yazdırmak için çarpma operatörünü kullanabilirsiniz.

```
print (20*'BA')
BABABABABABABABABABABABABABABABABABABABA
```

Bölme Operatörü

Bu operatör, iki sayıyı bölmek için kullanılır.

Örnek 30

```
print (5/3)
0.8333333333333334
```

Bölme operatörünü ikiden fazla sayıyı bölmek için de kullanabilirsiniz.

Örnek 31

```
print(18/3/2)
3.0
```

Bu tür durumlarda Python işlemleri soldan başlayarak sırayla yapar. Önce 18/3 işlemini yapar ve sonucu 2'ye böler.

Kuvvet Alma Operatörü

Bir sayının kuvvetini almak için kullanılır. Bir sayının kuvveti o sayının kendisiyle kuvvet kadar çarpılmasıyla bulunur. 5**3=5*5*5

Örnek 32

5'in 3. kuvvetinin bulunması:

```
print(5**3)
125
```

Kuvvet alma operatörü bir sayının kökünü almak için "1/kuvvet" olarak kullanılabilir.

Örnek 33

49'un karekökünü almak için 1/2 kuvveti alınır.

```
print(49**(1/2))
7.0
```

Tam Bölüm Operatörü

İki sayının birbirine tam bölüm sonucunu verir. Bölüm sonucu ondalıklı sayı ise ondalıklı kısmını almaz.

Örnek 34

```
print (121.00//3)
40.0
```

Mod Alma Operatörü

Bir sayının diğer bir sayıya bölümünden kalanını verir.

Örnek 35

```
print (5%3) #Sayının 3 ile bölümünden kalan.
print (9%2) #Kalan 0 ise sayı çifttir.
2
1
```

2.6.2. İlişkisel Operatörler

İlişkisel operatörler, değerler arasındaki ilişkiyi kontrol ederek sonucu "boolean" bir değer olarak (True, False) döndürür. "True" değeri şartın, ilişkinin veya koşulun sağlandığı anlamına gelirken, "False" değeri ise ilişkinin sağlanmadığı anlamına gelir.

İşaret	İşlem	Örnek	Sonuç
==	Eşit mi?	5==3	False
!=	Farklı mı?	5!=3	True
>	Büyüktür?	5>3	True
<	Küçüktür?	5<3	False
>=	Büyük veya eşittir?	3>=3	True
<=	Küçük veya eşittir?	5<=3	False
is	Değerler eşit mi?	'Elif' **is** 'elif'	False
is not	Değerler farklı mı?	'Elif' **is not** 'elif'	True
in	İçeriyor mu?	'bil' **in** 'bilişim'	True
not in	İçermiyor mu?	'bil' **not in** 'bilişim'	False

Eşittir Operatörü

"==" operatörü iki değerin birbirine eşit olup olmadığını anlamak için kullanılır. İki değer birbirine eşitse "True", eşit değilse "False" değeri verir.

Örnek 36

```
print(5==3)
#değişkenlere atadığınız değerleri de aynı şekilde kontrol edebilirsiniz.
sayi1=5
sayi2=3
print(sayi1==sayi2)
print(sayi1==5)
False
False
True
```

Örnek 37

"==" Operatörü karakter dizilerinde de değerlerin eşitliğini kontrol etmek için kullanılır.

```
print('Emre'=='emre')
# Küçük büyük harf duyarlılığına (case sensetive) dikkat ediniz.
metin1='Emre'
metin2='emre'
print(metin1==metin2)
print(metin1=='Emre')
False
False
True
```

Eşit Değildir Operatörü

"!=" operatörü iki değerin birbirinden farklı olup olmadığını anlamak için kullanılır. "==" operatörünün tersine değerler birbirine eşitse "False", eşit değilse "True" değerini döndürür.

Örnek 38

```
print(5!=3)
sayi1=5
sayi2=3
print(sayi1!=sayi2)
print(sayi1!=5)
# Çıktıların == operatörünün tersi olduğuna dikkat ediniz.
True
True
False
```

Örnek 39

"!=" operatörü karakter dizilerinde de değerlerin farklılığını kontrol etmek için kullanılır.

```
print('Emre'!='emre')
# Küçük büyük harf duyarlılığına (case sensetive) dikkat ediniz.
metin1='Emre'
metin2='emre'
print(metin1!=metin2)
print(metin1!='Emre')
True
True
False
```

Büyüktür Operatörü

Büyüktür “>” operatörü iki değeri karşılaştırmak için kullanılır. 1. sayı 2. sayıdan büyükse “True” değilse “False” değerini döndürür.

Örnek 40

```
sayi1=6.06
sayi2=6.07
print(sayi1>sayi2)
False
```

“sayi2” değişkenine yeni bir değer atandığını gözden kaçırmayınız.

```
sayi2=6
print (sayi1>sayi2)
sayi2=6.06
print(sayi1>sayi2)
True
False
```

Küçüktür Operatörü

Küçüktür “<” operatörü iki değeri karşılaştırmak için kullanılır. 1. sayı 2. sayıdan küçükse “True” değilse “False” değerini döndürür. Büyüktür operatörünün tersi işlevini görür.

Örnek 41

```
sayi1=6.06
sayi2=6.07
print(sayi1<sayi2)
True
```

Programımıza devam edelim. “sayi2” değişkenine yeni bir değer atadığımızı gözden kaçırmayın.

```
sayi2=6
print (sayi1<sayi2)
sayi2=6.06
print(sayi1<sayi2)
True
False
False
```

Karakter dizileri, alfabetik olarak sıralandığında sonra gelen ifade daha büyük olarak değerlendirilir.

```
print ('z'>'a')
print ('a'<'z')
True
True
```

Büyük Eşittir (>=) ve Küçük Eşittir Operatörleri

Büyük eşittir “>=” operatörü iki değeri karşılaştırmak için kullanılır. Birinci değer ikinciden büyükse veya ikinciye eşitse “True” değilse “False” değerini döndürür. Küçük eşittir “<=” operatörü ise birinci değer ikinci değerden küçükse veya ikinci değere eşitse “True” değilse “False” değerini döndürür.

Örnek 42

Büyük Eşittir (>=) ve Küçük Eşittir (<=) Operatörleri kullanımı:

```
sayi1=6.06
sayi2=6.06
print(sayi1>=sayi2)
print (sayi1<=sayi2)
#sayılar eşit olduğu için iki operatör de True değeri döndürür.
sayi2=6.07
print(sayi1>=sayi2)
print(sayi1<=sayi2)
#1.sayı 2. sayıdan küçük olduğu için sadece <= operatörü True değerini döndürür.
sayi2=6.05
print(sayi1>=sayi2)
print(sayi1<=sayi2)
#sayi2 değişkeni 6.05 olduğu ve sayi1 sayi2 den büyük olacağı için sadece >=
operatörü True değeri döndürür.
True
True
False
True
True
False
```

"is" ve "is not" Operatörleri

"is operatörü" ve "==" operatörü benzer işleve sahiptir ve iki değerin eşit olup olmadığını kontrol etmek için kullanılır. Değerler eşitse "True" değilse "False" değerini döndürür.

"is not" operatörü ise is operatörünün tersi işlev görür. "is" not operatörü değerler farklı ise True değerini değerler aynıysa "False" değerini döndürür.

NOT

"==" operatörü değerlerin eşitliğini kontrol ederken "is" aynı zamanda her iki değerin aynı nesneyi referans gösterip göstermediğini kontrol eder.

Örnek 43

```
sayi1=5
print (sayi1  is 5)
print (sayi1 is not 5) #is operatörünün tersini verir.
True
False
```

"is" operatörü karakter dizisinde de kullanılabilir.

```
print ('elif' is 'Elif')
#Yukarıdaki kod için büyük harf küçük harf duyarlılığını hatırlayınız.
adi='Elif'
print (adi is 'Elif')
print (adi is not 'Elif') #is operatörünün tersini verir.
False
True
False
```

"in" ve "not in" Operatörleri

"in" operatörü bir karakter dizisinin başka bir karakter dizisinde yer alıp almadığını kontrol etmek için kullanılır. Karakter dizisi, diğer karakter dizisi içinde yer alıyorsa "True" değeri, yer almıyorsa "False" değeri döndürür. "not in" operatörü ise içinde yer almıyorsa "True" yer alıyorsa "False" değeri döndürür.

Örnek 44

```
# in operatörü kullanımı
print ('Bil' in 'Bilişim')
# 2. karakter dizisi içinde 1. karakter dizisi var mı?
print ('Bil' not in 'Bilişim')
# in operatörünün tersi sonuç verir.
True
False
```

NOT

"in operatörünün" for döngüsünde işlevsel bir kullanımı vardır. "Döngüler" konusunda bu kullanımına yer verilmiştir.

2.6.3. Atama Operatörleri

Atama operatörleri değişkene değer atamak için kullanılır. Değişkeni başka bir değerle işleme alarak sonucun yine aynı değişkene atanmasını sağlar.

İşaret	İşlem	Örnek	Sonuç
+=	Artırarak atama	sayi1+=3	8
-=	Eksilterek atama	sayi1-=3	2
*=	Çarparak atama	sayi1*=3	15
/=	Bölerek atama	sayi1/=3	1.6666666666666667
* *=	Kuvvet alarak atama	sayi1**=3	125
//=	Tam sayı bölerek atama	sayi1//=3	1
%=	Mod alarak atama	sayi1%=3	2

NOT

sayi1=5 olarak atanmıştır.

Artırarak Atama Operatörü

Bir değişkenin üstüne sayısal değerleri toplayarak sonucun aynı değişkene atanmasını sağlar.

Örnek 45

Aşağıdaki kod ”sayi1=sayi1+3” koduyla aynı işlevi görür.

```
sayi1=5
sayi1+=3
print (sayi1)
metin1='Merhaba '
metin1+='Mars' #metin1=metin1+"Mars" koduyla aynı işlevi görür.
print(metin1)
8
Merhaba Mars
```

Eksilterek Atama Operatörü

Bir değişkenden bir sayısal değeri eksilterek sonucun aynı değişkene atanmasını sağlar.

Örnek 46

Aşağıdaki kod “sayi1=sayi1-3” koduyla aynı işlevi görür.

```
sayi1=5
sayi1-=3
print (sayi1)
2
```

Çarparak Atama Operatörü

Bir değişkeni bir sayısal değer ile çarparak sonucun aynı değişkene atanmasını sağlar.

Örnek 47

Aşağıdaki kod "sayi1=sayi1*3" koduyla aynı işlevi görür.

```
sayi1=5
sayi1*=
print (sayi1)
#Aynı şekilde karakter dizilerinde de kullanabilirsiniz.
metin1='Merhaba '
metin1*=3 #metin1=metin1*3 koduyla aynı işlevi görür.
print(metin1)
15
Merhaba Merhaba Merhaba
```

Bölerek Atama Operatörü

Bir değişkeni bir sayısal değere bölerek sonucun aynı değişkene atanmasını sağlar.

Örnek 48

Aşağıdaki kod "sayi1=sayi1/3" koduyla aynı işlevi görür.

```
sayi1=5
sayi1/=3
print(sayi1)
1.6666666666666667
```

Kuvvet Alarak Atama Operatörü

Bir değişkenin kuvvetini alarak sonucun aynı değişkene atanmasını sağlar.

Örnek 49

Aşağıdaki kod "sayi1=sayi1**3" koduyla aynı işlevi görür.

```
sayi1=5
sayi1**=3
print (sayi1)
125
```

Tam Sayı Bölerek Atama Operatörü

Bir değişken sayıya bölündüğünde sonucun (ondalık kısmını atarak) aynı değişkene atanmasını sağlar.

Örnek 50

Aşağıdaki "kod sayi1=sayi1//3" koduyla aynı işlevi görür.

```
sayi1=5
sayi1//=3
print (sayi1)
1
```

Mod Alarak Atama Operatörü

Bir değişkenin bir sayıya bölümünden kalanın aynı değişkene atanmasını sağlar.

Örnek 51

Aşağıdaki kod "sayi1=sayi1%3" koduyla aynı işlevi görür.

```
sayi1=5
sayi1%=3
print (sayi1)
2
```

2.6.4. Mantıksal Operatörler

İfadeleri mantıksal olarak bağlamak için kullanılan: “or”, “and” ve “not” operatörleridir.

“or” Operatörü

“or” operatörü “veya” anlamındadır. Belirtilen koşullardan birinin sağlanması durumunda “True” değeri döndürür. Bir sayı 6’dan küçük **veya** 10’dan büyükse koşulunu düşünelim. Sayımız 5 ise 6’dan küçük olduğu için bu şartı sağlayacaktır. Sayımız 6, 7, 8, 9 veya 10 olursa şartların her ikisini de sağlamadığı için “False” değeri döndürülür.

Örnek 52

```
sayi1=5
#or operatörü kullanımı
print (sayi1<6 or sayi1>10)
adi='Elif'
print (adi=='Elif' or adi=='Emre')
#Adı Elif veya Emre ise True değerini döndürür.
meslekUnvani='Mühendis'
print (meslekUnvani=='Öğretmen' or meslek=='Doktor')
#Meslek ünvanı Öğretmen veya Doktor olmadığı için False döndürür.
print (meslekUnvani=='Öğretmen' or meslekUnvani=='Doktor' or
meslekUnvani=='Mühendis')
#Meslek Ünvanı Öğretmen veya Doktor veya Mühendis'ten biri ise True değerini
döndürür.
# İkiden fazla koşul için de kullanılabilir.
True
True
False
True
```

"and" Operatörü

Bu operatör "ve" anlamındadır. Belirtilen koşulların hepsinin sağlanması durumunda "True" değerini döndürür. Bir öğrencinin ders puanı 50'den büyük ve 60'tan küçükse koşulunu düşünüldüğünde öğrencinin puanı 50 ise her iki şartı da sağlayacaktır ve "True" değerini döndürecektir.

Örnek 53

```
ogrenciDersPuani=50
print (ogrenciDersPuani>50 and ogrenciDersPuani<60)
adi='Emre'
yasi=24
print (adi=='Emre' and yasi>=20)
#Adı Emre ve yaşı en az 20 ise True değerini döndürür.
meslekUnvani='Mühendis'
askerlikDurumu='Yaptı'
isTecrubeYil=2
print (meslekUnvani=='Mühendis' and askerlikDurumu=='Yaptı')
print (meslekUnvani=='Mühendis' and askerlikDurumu=='Yaptı' and isTecrubeYil>=3)
False
True
True
False
```

Meslek unvanı Mühendis ise ve askerliğini yapmışsa "True" değerini döndürür.

Meslek ünvanı Mühendis ise, askerliğini yapmışsa ve en az 3 yıl iş tecrübesi varsa "True" değerini döndürür.

"not" Operatörü

Bu operatör "değil" anlamındadır. Belirtilen koşulun tersi doğruysa "True" değeri verir. Bir öğrencinin puanı 45'ten küçük değilse ifadesi düşünüldüğünde öğrencinin puanı 50 ise "True" değerini döndürür.

Örnek 54

```
ogrenciDersPuani=50
print(not(ogrenciDersPuani<45))
print (ogrenciDersPuani>45) #Yukarıdaki ifade ile aynı işlevi görür.
True
True
```

2.6.5. Operatörlerde Öncelik Sırası

Farklı operatörler birlikte kullanılabilir. Operatörler birlikte kullanılırken hangi işlemin önce yapılacağına dair bir sıralama vardır.

- Parantez içindeki işlemler her zaman öncelikli olarak yapılır.
- Çarpma ve bölme işlemleri toplama ve çıkarma işlemine göre önce yapılır.
- Aynı derecedeki operatörlerde işlem sırası önceliği soldan sağa doğrudur.
- İşlemlerde öncelik sırasını belirtmek için en iyi yöntem operatörleri parantez içinde kullanmaktır.

Örnek 55

```
print((3+5)*2) #Bu işlemin sonucunu tahmin ediniz.
#Öncelikle parantez içi yapıldığında 8*2=16
print (3+5*2) #Bu işlemin sonucu kaçtır?
#Öncelikle çarpma işlemi yapıldığından 3+10=13
print (3**2*2)
print (6*7/7)
print (6*3/2+8/2*3)
16
13
18
6.0
21.0
```

Bu bölümde temel operatörler verilmiştir. Python'da bu operatörlerin yanında başka operatörlerde yer almaktadır. Ayrıca kullandığınız Python sürümüne göre farklı operatörler kullanılabilir.

2.7. Bölüm Sonu Örnekleri

1. Aşağıdaki kod çalıştırıldığında ekrana "True" veya "False" değerlerinden hangisi yazılır?

```
sayi1=8*5/2
print (sayi1>20)
```

2. Aşağıdaki kod çalıştırıldığında ekrana "True" veya "False" değerlerinden hangisi yazılır?

```
vizePuani=80
finalPuani=70
sonuc=(0.4*vizePuani)+(finalPuani*0.6)
print (sonuc>=60)
```

3. Aşağıdaki değişken adlandırmalarından hangisi yanlıştır?

```
_sinav_puani1=80
sinavPuani=90
1sinavpuani=100
SınavPuanı=70
```

4. Aşağıdaki kod çalıştırıldığında ekrana hangi değişken tipi yazılır?

```
plaka='06'
print(type(plaka))
```

5. Aşağıdaki kod çalıştırıldığında ekrana ne yazılır?

```
#sehir='Ankara'
sehir='İstanbul'
print ('Şehir: ', sehir)
```

6. Aşağıdaki kod çalıştırıldığında ekrana ne yazılır?

```
sehir='Ankara'
sehir='İstanbul'
print (sehir!='istanbul') #operatöre dikkat
#Büyük harf küçük harf duyarlılığına dikkat
print (sehir=='İstanbul')
```

7. Veliye öğrencisinin devamsızlığını bildiren mesaj metni yazan kodları yazınız. Veli adını soyadını, mesajın içeriğini, devamsızlık süresini ayrı değişkenlere atayarak aşağıdaki gibi bir mesajda yazdırın. Sn..... .....öğrencinizin toplam devamsızlığı .... gündür. Python Lisesi

8. Tüketiciye su faturasını bildiren mesaj metni oluşturan kodları yazınız.

Kullanıcıya fatura tutarını belirten mesajda kullanılacak değişkenler: Hitap şekli, abone numarası, tüketim dönemi, tüketim tutarı*1.

Sayın ...nolu abonemiz ..... dönemi faturanız ..... TL'dir. Python Belediyesi

Cevaplar

1.
```
False
```

2.
```
True
```

3. "1sinavpuani" yanlıştır.
```
File "<ipython-input-35-36471b2a21d1>", line 3
1sinavpuani=100
^
SyntaxError: invalid syntax
```

4.
```
Karakter dizisi
<class 'str'>
```

5.
```
Şehir: İstanbul
```

6.
```
True
True
```

7.
```
onMesaj="Sayın "
veliAdi='Murat KARA'
mesajMetni =" öğrencinizin toplam devamsızlığı: "
devamsizlik='19.5'
veliMesaj=(onMesaj+veliAdi+mesajMetni+devamsizlik+' gün Python lisesi ')
print(veliMesaj)
Sayın Murat KARA öğrencinizin toplam devamsızlığı : 19.5 gün Python lisesi
```

8.
```
onMesaj="Sayın "
mesajSonu=" Mayıs 2020 dönemi faturanız: "
aboneNo="29101923"
tuketim=20
tuketimTutari=int(tuketim)*1.0
mesaj=onMesaj+str(aboneNo)+' nolu abonemiz ' +
mesajSonu+str(tuketimTutari)+" TL' dir. Python Belediyesi"
print(mesaj)
```
```
Sayın 29101923 nolu abonemiz  Mayıs 2020 dönemi faturanız : 20.0 TL' dir.
Python Belediyesi
```